Amendments to the Specification:

Please amend the paragraph starting on page 1, line 14 and ending on page 2, line 4 as follows:

A way of constructing error-correcting codes that are both powerful and can be efficiently decoded is to combine two or more smaller codes, called constituent codes, to form a larger composite code, with the decoding of the larger composite code being accomplished in an iterative manner using soft-in soft-out (SISO) decoding of the smaller constituent codes. The complexity of decoding a composite code is directly related to the complexity of SISO decoding the constituent codes. Single-parity-check (SPC) codes are amenable to low-complexity SISO processing because of their simple structure, and as a result, SPC codes are commonly used as constituent codes in composite code designs. The other type of code commonly used in composite code designs is convolutional codes. The regular trellis structure of convolutional codes results in SISO processing implementations of moderate complexity, provided that the number of states is small (e.g., 8 states). ~~It is interesting to note that an SPC code can in fact be considered to be a two-state systematic convolutional code, with a terminated (flushed) trellis, and parity puncturing.~~ Various adjectives are used in this document to describe data elements that are shared between constituent codes of a composite code, including "shared", "overlapped", and "interlocked".

Please amend the paragraph starting on page 3, line 12 and ending on page 4, line 5 as follows:

Another type of composite code structure using convolutional codes was proposed in: S. Benedetto, G. Montorsi, D. Divsalar, and F. Pollara, "Serial concatenation of interleaved codes: Performance analysis, design, and iterative decoding", JPL TDA Progress Report, vol 42-126, Aug. 1996. This report is incorporated herein by reference. The authors proposed serial concatenation of convolutional codes (SCCC). SCCC codes, like PCCC codes, consist of overlapping convolutional codes, but with SCCC, all of the (unpunctured) coded data elements of one convolutional code are provided as information data elements to a second convolutional code, with the second code being systematic and recursive ~~(of course, some of the coded data~~

~~elements of the first code may be punctured)~~. A disadvantage of SCCC, however, is that decoder hardware complexity is roughly doubled, on a per-state basis, as compared to PCCC. This is a direct consequence of the fact that with the first constituent code of an SCCC code, there are many state transition intervals where more than one data element is interleaved (or, more precisely, more data elements are interleaved than are necessary to determine the state transition). This means that in the iterative decoding of an SCCC code, soft-in soft-out processing needs to be able to provide soft outputs for not one but two data elements per state transition interval. As a result, non-local connectivity requirements, a dominant factor in implementation complexity, double. The fact that some state transition intervals have more than one interlocked (interleaved) data element is an inherent property of SCCC codes.

Please delete the paragraph on page 4, lines 6 to 20, which starts with "Yet another type".

Please amend the paragraph starting on page 4, line 21 and ending on page 5, line 7 as follows:

Turbo codes[[,]] and SCCC codes[[,]] ~~and H-TCC codes all~~ both use convolutional codes as constituent codes to form a composite code structure. The regular trellis structure of convolutional codes makes moderate-complexity SISO decoder implementations possible, and this is why composite code designs for iterative decoding have focussed on convolutional codes as constituent codes (SPC codes are also used as constituent codes in LDPC codes (low density parity check codes).[[,]] ~~but as mentioned earlier, these can be considered to be simple convolutional codes).~~ With convolutional constituent codes, however, there is an inevitable trade-off: either low-error rate performance is degraded, if only one data element per state transition interval is interlocked (interleaved), or decoder complexity is increased, if more than one data element per state transition interval is interlocked. That is, if only one data element per state transition is "overlapped", performance at low error-rates suffers. On the other hand, if two data elements per state transition are "overlapped", hardware decoder complexity is increased. This trade-off is inherent to the use of convolutional codes as constituent codes in composite code structures.

Please amend the paragraph starting on page 5, line 24 and ending on page 6, line 3 as follows:

A central feature of these codes is that state sequencing is driven not by source data alone, as is the case with convolutional codes, but rather by a sequence that includes both source data and so-called "inserted" data elements, the inserted data elements having a linear dependence on a ~~the state sequencing~~ state. In decoding, the consequence of the inserted data elements is that state transition intervals involving one or more inserted data elements are handled in a special way; ~~for state transition intervals with inserted data elements,~~ not all state transitions of the state sequencing are in fact possible, and so for such intervals the decoding considers only a reduced subset of the entire set of ~~state sequencer~~ state transitions.

Please amend the paragraph on page 8, lines 3 to 13 as follows:

Another embodiment of the invention provides a composite code encoder adapted to encode a sequence of source data elements to produce a first sequence of primary coded data elements which satisfy a first set of constraints of a first code equivalent to the encoder described above, and which after being re-ordered to form a second sequence of coded data elements, satisfy a second set of constraints of another code. The second set of constraints might be those of another code type entirely such as an RSC code for example, of a code type consistent with the first code (i.e., another injection code, but not necessarily of exactly the same form as the first injection code), or identical to those of the first code.

Please amend the paragraph starting on page 8, line 32 and ending on page 9, line 6 as follows:

While particular structures for encoders have been described, it is to be understood that, more generally, embodiments of the invention encompass any encoder adapted to implement a set of constraints equivalent to those of any of the encoders provided. For example, it would be possible to implement any of the above codes using a generator matrix structure, but the generator matrix structure would ultimately have to implement the same set of constraints.

Please amend the paragraph on page 12, lines 11 to 13 as follows:

The method might be adapted for example for application wherein the at least two constituent codes are exactly two constituent codes.

Please amend the paragraph on page 12, lines 14 to 23 as follows:

Embodiments of the invention also contemplate any of the above encoders or decoders[[,]] or early stopping methods[[,]] implemented as processing platform (application specific or general purpose) executable code stored on a processing platform readable medium. Furthermore, any of the above encoders or decoders[[,]] or early stopping methods may be implemented with any suitable combination of hardware and/or software. This might for example include programmable logic, firmware, software etc. implementations of encoders/encoding and decoders/decoding.

Please amend the paragraph on page 13, lines 17 to 19 as follows:

Figure 10 is a block diagram of an iterative decoder suitable for decoding the composite code type of which Figure 5 is an example; [[and]]

Please amend the paragraph on page 13, lines 20 to 21 as follows:

Figure 11 is a flowchart of a method of stopping an iterative decoder earlier than would normally occur[[.]] ; and

Please add the following paragraph on page 13, line 22:

Figure 12 is a diagrammatic illustration of a composite code in which a plurality of additional sets of constraints are applied.

Please amend the paragraph starting on page 17, line 15 and ending on page 18, line 2 as follows:

Figure 3 may be thought of as a block diagram showing components of a generalized injection code encoder provided by an embodiment of the invention, ~~a flowchart showing method~~ or as steps of an encoding method provided by an embodiment of the invention, ~~or as~~ and provides a definition of a code having a new structure provided by an embodiment of the invention. It is noted that Figure 3 does not provide the whole story of injection code generalization. An injection code is one having the structure of Figure 3, but also preferably constrained by a number of "injection code specifications" presented in detail below. For the purpose of description at this point, Figure 3 will be described from the point of view of an encoding method. The major steps include a data organization step generally indicated by 150, a linear state sequencing step generally indicated by 152, and a state-to-data-elements conversion step 155. The linear state sequencing step 152 will be described in detail further below, and for now it suffices to note that this step 152 maintains and outputs a state x_i 157. The functionality of the data organization step 150 and the state-to-data elements conversion 155 may be collectively referred to as a data insertion step or component.

Please amend the paragraph on page 32, lines 15 to 23 as follows:

For example, in one embodiment both the injection codes ~~might be~~ have a structure defined by the injection code encoder of Figure 1 which inserts a state-derived bit in every eighth sequencing bit location. Thus, every eighth bit of the first sequencing bit stream 400 is an inserted bit of the first injection code. Similarly, in the re-ordered sequencing bit stream 402, every eighth bit is an inserted bit of the second injection code. This is shown in Figure 5 by shading every eighth bit in both the first and second sequencing bit streams 400,402.

Please amend the paragraph starting on page 35, line 32 and ending on page 36, line 11 as follows:

To identify the parity check matrix for a single injection code requires examining the constraints associated with a single injection code. It can be seen in Figure 1 that the only "constrained" data elements are those at the inserted bit positions (recall that at this point in the discussion Figure 1 is being considered in isolation). Thus, there is a parity constraint associated with each inserted bit position. Fig. 1 shows that certain bit positions are "constrained", and others are not. This indicates that when creating an H matrix for a code having the code structure ~~as depicted in~~ produced by the encoder of Fig. 1, a constraint can be associated with each and every inserted bit position. Then it is simply a matter of determining the parity constraint for each such bit position.

Please add the following paragraph on page 56, line 1:

Referring now to Figure 12, shown is an example of a composite code encoder that applies a plurality of additional sets of constraints. The first set of constraints are illustrated generally by the "other" constituent code 1201. Two additional sets of constraints, represented by injection code 1203 and the "another" constituent code 1205 respectively are also shown. The constraints of these two other codes 1203, 1205 are applied to the primary coded data elements after respective re-orderings 1202 and 1204. While the example of Figure 12 is particular to two additional sets of constraints, more generally any number of additional constraints can be applied. A matrix representation of the encoder is shown generally at 1206.